REMARKS

I. INTRODUCTION

Claims 1-7 remain pending in the present application. No new matter added has been added. In view of the following remarks, it is respectfully submitted that all of the presently pending claims are allowable.

II. THE 35 U.S.C. § 103(a) REJECTIONS SHOULD BE WITHDRAWN

The Examiner has rejected claims 1-7 under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 6,529,985 to Deianov et al. (hereinafter "Deianov") in view of U.S. Patent No. 6,658,571 to O'Brien et al. (hereinafter "O'Brien").

Deianov describes a system for selective interception of system calls using system call wrappers to execute in a process address space of computer memory. (*Deianov*, col. 3, lines 30-34). Pointers to system calls that are to be intercepted and which are normally stored in an interrupt vector table are replaced with pointers to an interception module. (*Id.*, col. 3, lines 36-38). The interception module maintains an association table of the select processes and system call wrapper entry points. (*Id.*, col. 3, lines 58-62). The module determines from the association table whether the calling process is one of the selected processes with a registered entry point in a system call wrapper. If so, the interception module prepares to call the appropriate system call wrapper. (*Id.*, col 4, lines 5-13).

O'Brien discloses a security framework for wrapping software applications and restricting access to system resources. (*O'Brien*, Abstract). O'Brien uses a hyperviser, which is a layer of software disposed between hardware and an operating system that implements the same

instruction set as the hardware, within the the kernel but between the operating system and other software applications (*Id.*, col.2 line 62 - col. 3, line 1). The use of the hyperviser allows the computing system to "control access to computing resources . . . such as memory, files, network sockets and processes." (*Id.*, col. 3, lines 2-9). More specifically, the security framework disclosed in O'Brien loads security modules into the kernel which grant or deny access to computing resources based on either the application requesting access or the computing resource being requested. (*Id.*, col. 3, lines 38-45). Thus, each security module wraps the applications thereby preventing the applications' access to the resources unless authorized by the modules.

In contrast, claim 1 of the present application recites a method for providing either direct or indirect access to a software function by "*determining a current processing mode of an executing software function.*" The Examiner stated that Deianov does not teach or suggest "determining a current processing mode of an executing software function" as recited in independent claim 1 of the present application. To cure this deficiency the Examiner cited O'Brien. (08/21/04 Office Action, page 2, paragraph 5).

The present invention defines a processing mode as being either unprivileged or privileged. Unprivileged mode is "a mode of protection associated with lower level access rights to memory in the computing system." (*Specification*, page 5, paragraph 12). Privileged mode is "a mode of protection associated with higher level access rights to memory in the computer system." (*Id.*). A privileged processing mode is sufficiently trustworthy and is allowed direct access to the desired memory access, while an unprivileged mode is not sufficiently trustworthy, hence it is only allowed indirect access. Thus, regardless whether a software function is trustworthy or not (i.e., privileged or unprivileged) the function still has access to the requested

resource. The difference between the two modes is the type of access the software function obtains. Thus, the purpose of the examination of the "current processing mode" in the recited claim 1 is to determine the type of access the software function may obtain.

Conversely, O'Brien does not teach or suggest "determining a current processing mode of an executing software function" as recited in independent claim 1 of the present application. The Examiner referred to the exemplary embodiment disclosed in O'Brien using a web browser wrapped in a security module. (08/21/04 Office Action, page 3, paragraph 6; *O'Brien*, col. 7, lines 27-48). The security module enforces:

> [A] set of rules identifying which computing resources 106 the browser is allowed to access as well as what permissions the browser has. If there is no rule that allows access to a computing resource 106, then the security framework 101 refuses any requests to access that computing resource 106.

O'Brien, col. 7, lines 27-33.

Whether an application (e.g., web browser) is allowed to access a specific system resource based on a set of rules is different from the determination of "a current processing mode of an executing software function" as recited in independent claim 1. Specifically, reading preset rules or permissions is not a determination of a current processing mode. In the present application, the examination of the "current processing mode" is used to decide whether a software function should be allowed to access a system resource directly or indirectly and not whether it is allowed to access the resource or not. All of the software functions that are the object of the present application are already presumed to have access to system resources. Thus, this determination allows the computing system to decide the type of access that the software functions will obtain – direct or indirect.

In O'Brien access is controlled by security modules 105, therefore, it is these constructs that contain the preprogrammed rules set which are used to grant or deny access to the requesting applications. (*Id.*, col. 3, lines 38-45). O'Brien specifically states that the decision process of the security modules 105 is strictly based on: "(1) which application 107 is requesting access, or (2) which computing resource 106 is being requested." (*Id.*). Since the decision process of O'Brien is vastly different from the one recited in the present application and O'Brien is based on preset rules and permissions and not on a determination of a "current processing mode," O'Brien fails to cure the deficiency of Deianov. (*Id.*, col. 7, lines 27-33).

Neither Deianov or O'Brien include any showing or suggestion of a method for providing either direct or indirect access to software function comprising the steps of "*determining a current processing mode of an executing software function*" as recited in independent claim 1 of the present application. It is therefore respectfully submitted that claim 1 is not unpatentable over Deianov in view of O'Brien. Because claims 2-5 depend from and, therefore, include all of the limitations of claim 1, it is respectfully submitted that these claims are also allowable and the rejection of claims 1-5 under 35 U.S.C. § 103(a) should be withdrawn.

Independent claim 6 includes the same limitation as claim 1, i.e., "*determining a current processing mode of the program code segment.*" Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 6 is not unpatentable over Deianov in view of O'Brien

Independent claim 7 also includes the claim language "*determining a current processing mode of the program code segment.*" Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 7 is not unpatentable over Deianov in view of O'Brien.

IV. CONCLUSION

In light of the foregoing, the applicants respectfully submit that all of the now pending claims are in condition for allowance. All issues raised by the Examiner having been addressed, an early and favorable action on the merits is earnestly solicited.

Respectfully submitted,

Dated: October 25, 2004

By: _____
Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038